Docket No.: 20910/1206102-US1
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Letters Patent of:
Eric E. Lowe

Patent No.: 7,188,229

Issued: March 6, 2007

For: METHOD AND APPARATUS FOR MEMORY MANAGEMENT IN A MULTI-PROCESSOR COMPUTER SYSTEM

REQUEST FOR CERTIFICATE OF CORRECTION
PURSUANT TO 37 CFR 1.323 AND 1.322

Attention: Certificate of Correction Branch
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Upon reviewing the above-identified patent, Patentee noted typographical errors which should be corrected. A listing of the errors to be corrected is attached.

The typographical errors marked with an "A" on the attached list are found in the application as filed by applicant. Our check in the amount of $100.00 covering the fee set forth in 37 CFR 1.20(a) is enclosed.

The typographical errors marked with a "P" on the attached list are not in the application as filed by applicant. Also given on the attached list are the documents from the file history of the subject patent where the correct data can be found.

The errors now sought to be corrected are inadvertent typographical errors the correction of which does not involve new matter or require reexamination.

Transmitted herewith is a proposed Certificate of Correction effecting such corrections. Patentee respectfully solicits the granting of the requested Certificate of Correction.

The Commissioner is authorized to charge any deficiency of up to $300.00 or credit any excess in this fee to Deposit Account No. 04-0100.

Dated: March 30, 2007

Respectfully submitted,

By [illegible]
Flynn Barrison
Registration No.: 53,970
DARBY & DARBY P.C.
P.O. Box 5257
New York, New York 10150-5257
(212) 527-7700
(212) 527-7701 (Fax)
Attorneys/Agents For Applicant

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,188,229 Page 1 of 1

APPLICATION NO.: 10/769,586

ISSUE DATE : March 6, 2007

INVENTOR(S) : Lowe

It is certified that an error appears or errors appear in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On Sheet 2 of 11, in FIG. 2.1, in "Box 203", line 3, delete "idenitifier" and insert - - identifier - -, therefor.

In column 1, line 11, delete "Eric F. Lowe," and insert - - Eric E. Lowe, - -, therefor.

In column 7, line 33, delete "TTE' S." and insert - - TTE's. - -, therefor.

In column 19, line 27, in Claim 17, delete "Unites" and insert - - Units - -, therefor.

In column 19, line 28, in Claim 17, delete "lookside" and insert - - lookaside - -, therefor.

In column 17, line 43-44, in Claim 10, delete "a miss exception event comprises"

MAILING ADDRESS OF SENDER (Please do not use customer number below):
John W. Branch, Esq.
DARBY & DARBY P.C.
P.O. Box 5257
New York, New York 10150-5257

Issued Patent Proofing Form File#: **20910/1206102-US1**
Note: P = PTO Error A = Applicant Error

US Serial No.: **10/769,586** US Patent No.: **US 7,188,229 B2** Issue Dt.: **Mar. 6, 2007**
Title: **METHOD AND APPARATUS FOR MEMORY MANAGEMENT IN A MULTI-PROCESSOR COMPUTER SYSTEM**

Sr. No.	**P/A**	**Original**		**Issued Patent**		**Description Of Error**
		Page	**Line**	**Column**	**Line**	
1	A	Sheet 2 of 11 Drawings (03/29/2004)	3 (Box 203) (FIG. 2.1)	Sheet 2 of 11 (Box 203) (FIG. 2.1)	3	Delete "idenitifier" and insert - - identifier - -, therefor.
2	P	Page 2 of 16 Request for Corrected Filing Receipt (12/05/2006)	4 (Amendments to the specification)	1	11	Delete "Eric F. Lowe," and insert - - Eric E. Lowe, - -, therefor.
3	P	Page 11 Specification (01/30/2004)	13	7	33	Delete "TTE' S." and insert - - TTE's. - -, therefor.
4	A	Page 10 of 14 Claims (09/21/2006)	Claim 35 Line 2	19	27	In Claim 17, delete "Unites" and insert - - Units - -, therefor.
5	A	Page 10 of 14 Claims (09/21/2006)	Claim 35 Line 3	19	28	In Claim 17, delete "lookside" and insert - - lookaside - -, therefor.
6	A		Claim 10	17	43-44	Delete "a miss exception event comprises"

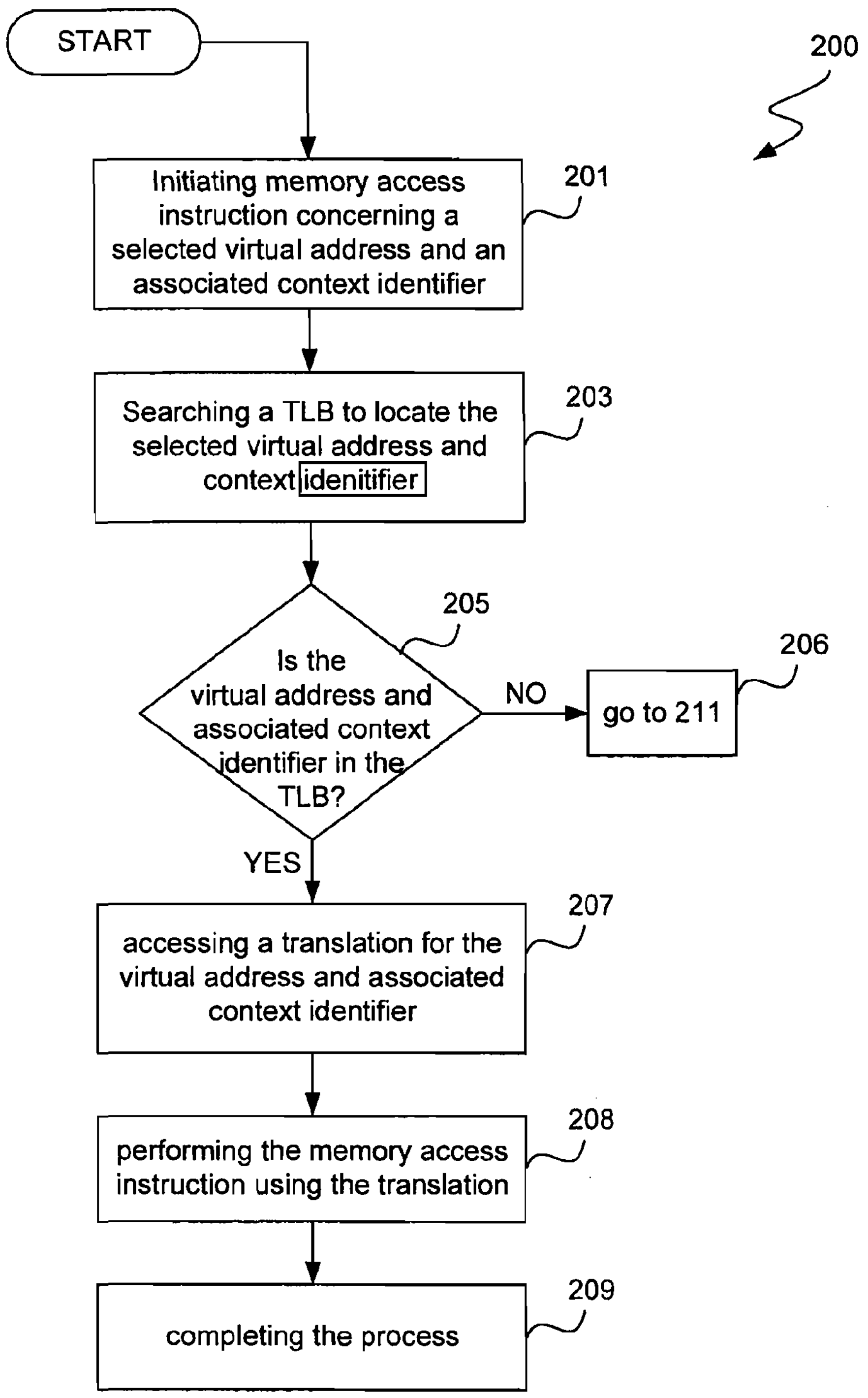
START
200
Initiating memory access instruction concerning a selected virtual address and an associated context identifier
201
Searching a TLB to locate the selected virtual address and context idenitifier
203
Is the virtual address and associated context identifier in the TLB?
205
NO
go to 211
206
YES
accessing a translation for the virtual address and associated context identifier
207
performing the memory access instruction using the translation
208
completing the process
209

Fig. 2.1

METHOD AND APPARATUS FOR MEMORY MANAGEMENT IN A MULTI-PROCESSOR COMPUTER SYSTEM

RELATED APPLICATION

The invention described herein relates to and claims priority from the Provisional Patent Application No. 60/537,431, entitled "Method and Apparatus for Memory Management in a Multi-Processor Computer System", invented by Eric F. Lowe, filed on Jan. 17, 2004. The aforementioned patent document is hereby incorporated by reference in its entirety for all purposes.

BACKGROUND OF THE INVENTION

The present invention relates to multiprocessor computing systems and, more particularly, to memory management unit (MMU) trap synchronization in multiprocessor computing systems. Multiprocessor computing systems are coming into increasingly common usage due to the many advantages inherent in such systems. In such systems, a single operating system controls the operation of all the microprocessors (CPU's) of the system. Common multiprocessor systems include scalable shared memory (SSM) system symmetric multiprocessor (SMP) systems. System symmetric multiprocessing can make use of multiprocessor computing architectures configured so that all CPU's can access all random access memory locations. Many architecture can implement such systems. Example include without limitation X86 systems as well as SSM system symmetric multiprocessor system architectures designed by Motorola, IBM, and Microsoft (e.g., NT systems). Linux based systems can also take advantage of the principles of the invention. Another architecture suitable for implementing the principles of the invention is the so-called SPARC architecture. SPARC is short for Scalable Processor Architecture, a RISC (reduced instruction set computer) technology developed by Sun Microsystems. The term SPARC® itself is a trademark of SPARC International, an independent organization that licenses the term to Sun for its use. The details of the SPARC specification are well known to persons having ordinary skill in the art and can be found in many standard references. One example of such a reference is entitled "The SPARC Architecture Manual, Version 9", by SPARC International Edited by David Weaver and Tom Germond, which is hereby incorporated by reference.

Scalable shared-memory multiprocessors distribute memory among the many processors of a system and use scalable interconnection networks to provide high bandwidth and low latency communication. In addition, memory accesses are cached, buffered, and pipelined to bridge the gap between the slow shared memory and the fast processors. Unless carefully controlled, such architectural optimizations can cause memory accesses to be excessively concentrated in the slow shared memory rather than taking advantage of the high speed memory contained in the processors.

As is known to persons having ordinary skill, memory management units (MMU's) maintain listings that include mappings from virtual addresses to associated physical addresses (which exist in RAM). The advantage of such mappings is that the more commonly used physical addresses can be stored in a microprocessor cache for high-speed access. On the downside is the fact that the cache memory used to store these mappings is very small. Such mappings include a physical address and virtual address

(also referred to as a translation table entry or TTE) a listing of attributes (such as memory access protection attributes), and a context identifier (which SPARC refers to as a MMU context). Such information is commonly stored in a translation lookaside buffer (TLB) (also known as TB(2), translation buffer, ATC, or address translation cache). The TLB is a small piece of associative memory within a processor which caches part of the translation from virtual addresses to physical addresses. Thus, whenever physical address information is required, the TLB is consulted. One significant advantage of the cache memory is that it performs operation extremely quickly. Thus, it is desirable to take advantage of such cache memory as much as possible.

However, whenever a required translation for a particular virtual address is not present in the TLB the required information must be acquired from another memory asset, which is commonly much slower. This process is referred to as a "TLB miss trap" or alternatively as a type of "MMU trap". In such cases ("miss traps") the address translation must be resolved using other mechanisms. For example, these translations from virtual to physical addresses (as well as other associated information) are also stored in secondary memory resources (also referred to herein as secondary memory assets. Commonly, the secondary memory assets include translation storage buffers (TSB's) and page tables (which stores the entire virtual address space description of each process). These secondary memory assets are commonly located in random access memory (RAM) where they are less easily accessed. Accessing virtual addresses from such secondary resources is a much slower process than TSB access and commonly slows down the system operation. This is especially problematic when remappings of the virtual address space require frequent changes to the TLB.

An additional difficulty encountered when virtual addresses or virtual address spaces are being remapped is that other processes or threads seeking access to the affected virtual addresses cannot access the information without causing inconsistent or fatal results. Therefore, in a multiprocessor computer system, maintaining a consistent view of a virtual address space in a multi-threaded process is critical. To obtain this consistent view requires synchronization by the operating system kernel across all of the processors in the system whenever a process virtual address space changes. Thus, when virtual addresses are changed they must be changed for all processes and all processors across the entire system (this process is referred to as synchronization). Such an operation requires that the old translation table entry for a given virtual address be removed (called "demapping" or "unmapping") and a new translation table entry (having a different physical address) be entered. This is referred to a "TLB shootdown". Additionally, all TLB shootdown events must be synchronized across the entire virtual address space for all CPU's in the system. Such synchronization prevents inconsistent mappings so that no single virtual address maps to more than one different physical address. Thus, the TLB's of all CPU's for any given virtual address all map to the same translation table entry.

Further, a difficulty arises when virtual address is being unmapped and another process (possibly running on another processor) seeks to access the same shared memory resource (e.g., a translation table entry (TTE) shared by another process or thread). Such activity can lead to an inconsistent view of the virtual address space. In such a situation the multiple threads will get inconsistent views of the process virtual address space while the virtual address space is changing.

IBM, and Microsoft (e.g., NT systems). Linux based systems can also take advantage of the principles of the invention. As explained above, SPARC architectures are suitable for implementing the principles of the invention. One example of a SPARC® compliant system is a Solaris® based system.

Each CPU **101** includes a memory cache configured to include a translation lookaside buffer (TLB) **102** having entries configured to include virtual addresses and associated context identifiers for those virtual addresses as well as physical address translations and attribute information. The system includes secondary memory assets **110** that can include translation storage buffers (TSB's) **111** and page tables **112**. Such secondary resources are typically stored in random access memory (RAM) but can also be stored as disk memory or at other memory locations. As is known to persons having ordinary skill in the art, a TSB is a specific location in memory specifically dedicated to handle the virtual address space for a specified process (or thread). Because this location is always known and densely packed, memory access to the virtual address space of such TSB's is quicker than for other RAM memory. Such TSB's **111** and page tables **112** can each include many translation table entries (TTE). Each TTE includes translation information concerning a virtual address and associated context identifier. Each TTE includes a physical address translation and attribute information for each corresponding virtual address and context identifier. Other information associated with the TTE's can also be stored in other memory, for example, virtual address space identifiers (VASI)(also commonly referred to as HAT pointers). Information stored in the VASI can include the "busy" or "lock" protection status of a range of TTE' S. One difficulty with information stored in a VASI is that it is not stored close to the TTE. Therefore, it can take a significant amount of time (in computer processing terms) to access this information. Consequently, the less access to the VASI information is required, the faster a system can run.

As previously indicated, the MMU **121** includes a TLB miss handler **122** and a miss exception handler **123**.

The TLB miss handler is a piece of software code configured to facilitate corrective action when a TLB miss occurs (i.e., when a memory access instruction accesses a virtual address that has no physical address translation currently resident in the TLB or when for some reason the required translation information cannot be found in the TLB). When a TLB miss occurs, the TLB miss handler **122** is invoked. The TLB miss handler **122** tests the context identifier associated with the virtual address to determine if a TTE (having the necessary translation information) is available to have a memory access instruction executed thereon. Some embodiments make advantageous use of this feature to perform context testing before the searching of the secondary memory assets for the TTE. Thus, the availability/unavailability of the TTE can be assessed prior to searching secondary memory assets. This can significantly increase system efficiencies. The availability or unavailability of TTE entries will be treated more fully hereinbelow. Once it has been determined that a selected TTE is available, the TLB miss handler **122** searches secondary memory assets **110** (e.g., the TSB's **111** and page tables **112**) to find the selected TTE having the necessary translation information. The selected TTE is related to the virtual address (and its associated context identifier) that is the subject of the TLB miss and typically contains the required physical address translation information as well as other information (e.g., attribute information) concerning the virtual address.

In cases where context testing reveals that selected TTE is unavailable, the TLB miss handler **122** invokes a miss exception handler **123** to facilitate the resolution of the TLB miss exception that has rendered the TTE unavailable. The exception handler determines the nature of the TLB miss exception that has rendered the TTE unavailable. Based on this determination, the exception handler **123** operates to resolve the unavailability. The exception handler **123** can determine that the miss exception has been resolved, resolve the miss exception, or it can selectively pause the operation of the exception handler **123**, if needed, until the miss exception is resolved and the TTE becomes available to have memory access instructions performed on it again.

FIG. **2** illustrates an exemplary flow diagram that describes a method **200** embodiment used to conduct memory operations in a multi-processor computer in accordance with the principles of the invention. The method can be used by any multi-processor system where the CPU's are controlled by a single operating system. A non-exclusive example of such a system is illustrated by FIG. **1**. The method **200** can be used by many types of multiprocessor systems including, but not limited to, SPARC® compliant systems (e.g., a Solaris® based system) as well as other multi-processor systems. The method **200** can be used by SPARC® compliant systems (e.g., a Solaris® based system) as well as other multi-processor systems. In such a system a memory access instruction is initiated (Step **201**). The instruction is to be executed using a selected virtual address and requires a physical address translation for the virtual address and its associated context identifier. The translation lookaside buffer (TLB) is searched to find the selected virtual address and its associated context identifier (Step **203**). A determination is made as to whether the selected virtual address and its associated context identifier are found in the TLB (Step **205**).

Where the selected virtual address and its associated context identifier are located in the TLB, a translation for the selected virtual address and its associated context identifier are accessed (Step **207**). The translation commonly includes the physical address of the desired physical memory resource sought. Additionally, the translation can include attribute information and other types of information. Once the translation is accessed, the memory access instruction (initiated in Step **201**) is now performed using the translation (Step **208**). This completes the process (Step **209**).

Where the selected virtual address and its associated context identifier are not located in the TLB, the method tests the context identifier that is associated with the selected virtual address (Step **211**). Testing the context identifier at this stage (prior to searching the secondary memory assets) allows an "up front" determination as to whether a sought after translation table entry (TTE) located in the secondary memory assets is available to have memory access instructions performed on it. Thus, an availability determination can be made prior to the time consuming search operations required to find the TTE in the secondary memory assets.

A determination is made as to whether the sought after TTE (i.e., the TTE that includes the translation information for the selected virtual address and context identifier) is the subject of a miss exception event or not (Step **213**). If the sought after TTE is the subject of a miss exception event (the nature of which will be described in greater detail below) it is considered unavailable to have memory access instructions performed thereon. Otherwise the TTE is available to have memory access instructions performed thereon.

Where testing the context identifier reveals that the sought after TTE is available to have memory access instructions wherein resolving the unavailability includes:

determining the nature of the unavailability; and

where it is determined that the cause of the unavailability is a demapping operation for a TTE that is shared by the virtual address for which the translation is sought,

pausing until the TTE demapping operation is completed; and

returning to initiating an access instruction.

9. A method of accomplishing memory management of miss exceptions in a memory management unit of a multi-processor computer system, the method comprising;

determining that a miss exception event has occurred, wherein the miss exception event concerns one of: an unassigned context identifier event, a memory access event changing a shared memory resource, and a translation storage buffer (TSB) resizing event;

resolving the miss exception event in accordance with a miss event resolution protocol suitable for resolving the received miss exception event; and

wherein said determining determines that the miss exception event comprises an instruction to change a translation table entry (TTE) that is shared by more than one virtual address space wherein each virtual address space has an associated context identifier; and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

identifying virtual address spaces that share the same TTE;

activating a lock for each virtual address space that shares the same TTE to prevent other processes from accessing the TTE while the lock is activated;

changing the corresponding context identifier for each virtual address space that shares the same TTE thereby making the associated TTE unavailable to have memory access instructions performed thereon;

performing the changes on the TTE;

releasing the locks on each locked virtual address space; and

freeing the corresponding context identifiers for each affected virtual address space.

10. The method of claim 9 wherein said determining determines that the miss exception event comprises ~~a miss exception event comprises~~ an instruction to resize a translation storage buffer (TSB)); and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

activating a lock for the TSB to prevent other processes from accessing entries in the TSB while the lock is activated;

changing the corresponding context identifier to indicate that the TSB virtual address space is unavailable to have memory access instructions performed thereon;

resizing the TSB;

changing the corresponding context identifier back to its original configuration; and

releasing the lock on the TSB.

11. The method of claim 9 wherein said determining determines that the miss exception event comprises an instruction concerning an unassigned context identifier; and

wherein resolving the miss exception event in accordance with a miss event resolution protocol comprises:

assigning a context identifier for a virtual address space associated with a TSB; and

assigning a portion of memory to the TSB.

12. A recordable computer readable medium including computing program code for handling translation lookaside

buffer (TLB) miss exceptions in a memory management unit of a multi-processor computer system having a plurality of Central Processing Units (CPU's), wherein said computer readable medium comprising:

computer program code for initiating an access instruction;

computer program code for determining that a TLB miss exception event has occurred, wherein determining that a TLB miss exception event has occurred includes testing a context identifier for the affected virtual address to determine whether a TLB miss exception has occurred;

computer program code for invoking a miss exception handler;

computer program code for determining the nature of the TLB miss exception event;

computer program code for resolving the miss exception event;

computer program code for returning to initiating an access instruction;

wherein determining the nature of the TLB miss exception event includes testing the context identifier for the affected virtual address to determine whether the TLB miss exception event results from one of an unassigned context identifier, a translation storage buffer (TSB) resizing operation that affects the virtual address, and an unmapping of a shared translation table entry (TTE) that is associated with the virtual address wherein the TTE comprises a shared memory resource;

wherein resolving the miss exception events includes resolving each type of miss exception event in accordance with specified miss exception resolution protocol for each type of miss exception event.

13. A computer readable medium as recited in claim 12, wherein resolving the unassigned context identifier miss exception event further includes the steps of:

assigning a context identifier value to a virtual address space that is to be associated with a process that contains the virtual address;

assigning selected portion of memory to a TSB that is to be associated with the virtual address space having the assigned context identifier;

updating at least one of secondary memory assets and the TLB with TSB and context identifier information; and

returning to initiating an access instruction.

14. A computer readable medium as recited in claim 12, wherein said shared memory resource comprises a translation table entry (TTE).

15. A computer readable medium as recited in claim 12, wherein resolving the miss exception event further includes the steps of:

determining if the shared memory resource is locked;

if the shared memory resource is not locked,

returning to initiating an access instruction; and

if the shared memory resource is locked,

pausing the exception handler until the shared memory resource becomes unlocked; and

returning to initiating an access instruction when the shared memory resource becomes unlocked.

16. A computer readable medium as recited in claim 12, wherein resolving the miss exception event further includes the steps of:

A) determining if the virtual address space of the TSB undergoing resizing is locked;

1) in a case wherein the virtual address space of the TSB undergoing resizing is locked,

i) pausing the exception handler until the virtual address space of the TSB undergoing resizing becomes unlocked;
ii) locking the virtual address space of the TSB undergoing resizing;
2) in a case wherein the virtual address space of the TSB undergoing resizing is unlocked,
i) locking the virtual address space of the TSB undergoing resizing;
B) once the virtual address space of the TSB undergoing resizing has been locked by one of 1)(ii) and 2(i), determining whether the TSB undergoing resizing has been assigned a specified location in memory;
1) if the TSB undergoing resizing has been assigned a specified location in memory,
i) releasing the lock on the virtual address space of the TSB undergoing resizing; and
ii) returning to initiating an access instruction;
2) if the TSB undergoing resizing has not been assigned a specified location in memory,
i) assigning a specified portion of memory to the TSB undergoing resizing;
ii) releasing the lock on the virtual address space of the TSB undergoing resizing; and
iii) returning to initiating an access instruction.

17. A multi-processor computing system having a plurality of Central processing $\boxed{\text{Unites}}$ (CPUs), wherein said multi-processor is operable for handling translation $\boxed{\text{lookside}}$ buffer (TLB) miss exceptions in a memory management unit of said multi-processor computer system, and wherein said multi-processor computing system is further operable to:
initiating an access instruction;
determining that a TLB miss exception event has occurred, wherein determining that a TLB miss exception event has occurred includes testing a context identifier for the affected virtual address to determine whether a TLB miss exception has occurred;
invoking a miss exception handler;
determining the nature of the TLB miss exception event;
resolving the miss exception event; and
returning to initiating an access instruction,
wherein determining the nature of the TLB miss exception event includes testing the context identifier for the affected virtual address to determine whether the TLB miss exception event results from one of an unassigned context identifier, a translation storage buffer (TSB) resizing operation that affects the virtual address, and an unmapping of a shared translation table entry (TTE) that is associated with the virtual address wherein the TTE comprises a shared memory resource;
wherein resolving the miss exception events includes resolving each type of miss exception event in accordance with specified miss exception resolution protocol for each type of miss exception event.

18. A multi-processor computing system as recited in claim **17**, wherein said testing the context identifier determines that said unavailability results from a miss exception due to an unassigned context identifier, and
wherein resolving the unassigned context identifier miss exception event further includes the steps of:
assigning a context identifier value to a virtual address space that is to be associated with a process that contains the virtual address;
assigning selected portion of memory to a TSB that is to be associated with the virtual address space having the assigned context identifier;

updating at least one of secondary memory assets and the TLB with TSB and context identifier information; and
returning to initiating an access instruction.

19. A multi-processor computing system as recited in claim **17**, wherein said testing the context identifier determines that said unavailability results from a miss exception due to a situation wherein the affected virtual address maps to a shared memory resource, and
wherein resolving the miss exception event further includes the steps of:
determining if the shared memory resource is locked;
if the shared memory resource is not locked,
returning to initiating an access instruction; and
if the shared memory resource is locked,
pausing the exception handler until the shared memory resource becomes unlocked; and
returning to initiating an access instruction when the shared memory resource becomes unlocked.

20. A multi-processor computing system as recited in claim **17**, wherein testing the context identifier determines that said unavailability results from a miss exception due to a situation wherein the virtual address is associated with a TSB that is undergoing a resizing operation; and
wherein resolving the miss exception event further includes the steps of:
A) determining if the virtual address space of the TSB undergoing resizing is locked;
1) in a case wherein the virtual address space of the TSB undergoing resizing is locked,
i) pausing the exception handler until the virtual address space of the TSB undergoing resizing becomes unlocked;
ii) locking the virtual address space of the TSB undergoing resizing;
2) in a case wherein the virtual address space of the TSB undergoing resizing is unlocked,
i) locking the virtual address space of the TSB undergoing resizing;
B) once the virtual address space of the TSB undergoing resizing has been locked by one of 1)(ii) and 2(i), determining whether the TSB undergoing resizing has been assigned a specified location in memory;
1) if the TSB undergoing resizing has been assigned a specified location in memory,
i) releasing the lock on the virtual address space of the TSB undergoing resizing; and
ii) returning to initiating an access instruction;
2) if the TSB undergoing resizing has not been assigned a specified location in memory,
i) assigning a specified portion of memory to the TSB undergoing resizing;
ii) releasing the lock on the virtual address space of the TSB undergoing resizing; and
iii) returning to initiating an access instruction.

21. A multi-processor computing system of accomplishing memory management of miss exceptions in a memory management unit of a multi-processor computer system, the method comprising;
determining that a miss exception event has occurred, wherein the miss exception event concerns one of: an unassigned context identifier event, a memory access event changing a shared memory resource, and a translation storage buffer (TSB) resizing event;
resolving the miss exception event in accordance with a miss event resolution protocol suitable for resolving the received miss exception event; and